REMARKS

I. Status of the Claims:

Claims 1-30 are currently pending. By this Amendment, claims 1-11 and 13-30 have been amended, and claim 12 has been canceled without prejudice or disclaimer. No new matter has been introduced by this Amendment. Upon entry of this Amendment, claims 1-11 and 13-30 would be pending.

II. Rejection Under 35 U.S.C. § 112:

Claims 2-18 and 20-30 are rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter.

A. Claims 2-12, 16 and 26-29:

The Examiner has rejected these claims particularly in regard to the preamble language "a method". Per the Examiner's suggestions, these claims have been amended to reflect in the preamble --the method--.

B. Claims 20-25:

The Examiner has rejected these claims particularly in regard to the preamble language "a distributed system". Similar to the Examiner's suggestions noted above, these claims have been amended to reflect in the preamble --the distributed system--.

C. Claims 13-15, 17, 18 and 30:

The Examiner has rejected these claims particularly in regard to whether these claims are dependent or independent and/or the language of "in anyone of the preceding claims". These claims have been amended to address the Examiner's concerns.

In view of the foregoing, the claims are believed to be definite and thus reconsideration and withdrawal of these rejections are respectfully requested.

III. Rejection Under 35 U.S.C. § 103:

A. Claims 1 to 5, 12 to 18, 26 to 30:

Claims 1-5, 12-18, 26-30 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kazi et al.

Both independent claims 1 and 13, as amended, are directed to methods involving translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor; transmitting the virtual processor code from a server to a client device; and translating the virtual processor code into native code which uses an instruction set of a physical processor of the client device. As further claimed, the bytecode is stack-based, and the virtual processor code is register-based.

For each of the reasons set out below, the Applicant respectfully submits that the Examiner has failed to establish *a prima facie* case of obviousness of these claimed inventions in light of Kazi. See MPEP .§ 2142.

First, as required by MPEP §§ 2142, 706.02 (J) and 2143, the prior art reference must teach or suggest all the claim limitations. The Examiner states that *"one simply implements a dawn line as a connection between two elements, transmitter and receiver". This* would imply that the elements to the left of the *"native machine code"* box in Figure 2 of Kazi are located on

a server, that the arrows pointing into the native machine code block correspond to a transmission over a network and the *"native machine"* and *"target processor"* are located on the client device. There is simply no disclosure of this whatsoever, in Kazi. Nor is there any teaching which would suggest that this could or would be the case. Accordingly, Kazi fails to teach or suggest all the claim limitations.

Second, the Examiner states that *"it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include networking"*. However, as provided in MPEP § 2142, the fact that the claimed invention is within the capabilities of one of ordinary skill in the art is not sufficient by itself to establish *prima facie* obviousness. What would be required is some objective reason to combine the teaching of the prior art to arrive at the claimed invention. Such an objective reason has not been made out by the Examiner. As stated in MPEP § 2143.01, the mere fact that a reference can be modified does not render the resulting combination obvious unless the prior art also suggests the desirability of the combination. There must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. This has not been made out by the Examiner.

Third, obviousness cannot be established with art that teaches away. In evaluating obviousness it is not appropriate to selectively consider part of a reference while ignoring other parts that teach away from the invention. As stated in MPEP § 2141.02, a prior art reference must be considered in its entirety, i.e. as a whole, including portions that would lead away from the claimed invention. It is stated throughout Kazi that the intermediate representations are *"internal"* to the compiler. They are thus clearly **not** transmitted via the networks. For example, see page 14, second full paragraph starting *"caffeine..."*, third line and

page 15, first full paragraph starting *"the native executable translation..."*, line 8. In as far as Figure 2 relates to JIT compilers, these compile bite code at run time i.e. at the client device if such a device were mentioned at all (see page 9, first paragraph of heading 3.2.2, lines 2 to 4). If at all, it is therefore clearly the byte code which should be transmitted according to Kazi and not any intermediate representation.

Fourth, the Examiner states that networking *"has been used commonly, as a means for sending independent platform intermediate code to another back end remotely for providing targeted translations because growing of many different processor types"*. This is a mere assertion and the Examiner provides no evidence that this is indeed the case. In the event that the Examiner continues such assertion, the Applicant respectfully requests that the Examiner provide objective evidence in support of this assertion.

Fifth, in any event, it is noted that even if one of ordinary skill in the art, starting from Kazi, tries to implement a system in which an intermediate representation is sent over a network and then compiled at the client device, the teaching of Kazi would not actually enable him to do so for the following reasons.

The intermediate codes referred to in Kazi cannot simply be transmitted over the network because they are internal products of the compiler (see above). Although Kazi is a review of many different state of the art compilers, it fails to show how these internal products could be changed to send them over a network, nor indeed is there any indication that this could or has been done. The Java bytecode class file in Figure 2 includes in addition to the byte code instructions for the methods of the class data which allows methods of the class to reference each other and to be referenced by methods of other classes (see page 12, *"constant pool"*, *"methods ??"* and information relating the class to other classes). This information of the class files is

used to fix the references in the methods of the class such that they can be resolved when the resulting native code is executed. This happens either at run time in a java virtual machine or on compilation into native code in direct compilers.

If one were to take the internal representations referred to in Kazi and transmit them over a network for compilation into native code, the resulting system would not work because the claimed device would have no information on how to resolve the references in the intermediate language. There is no suggestion in Kazi of how this could be achieved and therefore, even if one of ordinary skill in the art would attempt to modify the system disclosed in Kazi as suggested by the Examiner, such a person would end up with a system which does not work because there is no disclosure in Kazi how the intermediate representations could be turned into native code once they have been transmitted to the client device.

Finally, it is noted that the Examiner refers to the *"byte code translator"* in Figure 2 of Kazi to read on the feature (a) of claim 1, that is translating into virtual processor code. However, the virtual processor code is now specified to be register-based whereas the output of the byte code translator in Figure 7 is source code of a high level language such as C (see section 3.2.3 of Kazi) and not a register-based representation which resembles the architectural native processors.

For each of the reasons set out above, claims 1 and 13 and their dependent claims are believed to be distinguishable over the cited reference. Reconsideration and withdrawal of the rejection of these claims are respectfully requested.

B. Dependent Claims 6 to 10:

Claims 6-11 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kazi et al. in view of Miller et al. The Applicant respectfully traverses this rejection for the reasons set forth below.

These claims relate to *"fixing up"* the byte code when it is translated into virtual processor code, which means that instructions are provided which allow the references to a field within the class to be located by the native code (see claim 5) or the reference to another class or field or method in another class to be resolved (see claim 10). See also page 11 line 26, page 17 line 4 for a detailed description of these topics. By contrast, the disclosure of Miller, and in particular the portions referred to by the Examiner, relate to a programming language which has size-indefinite variables and its compilation into native code, including generating a size-definite variable corresponding to the size-indefinite variable according to machine specific criteria. The portion cited by the Examiner relate either to turning an intermediate language into native code, generally (for example column 1 lines 15 to 20, column 6 lines 1 to 10), or the determination of the size of the variable in the native code (column 7 line 20 to 30 and 37 to 46, column 6 lines 30 to 36, for example). There is no disclosure of resolving references within or between classes, let alone any reference to *"fixing* up" as described above. Accordingly, Miller cannot make up for the shortfall in Kazi and even a combination of these two references fails to teach or suggest all the elements of the respective claims.

Further, it is noted that any combination (which would still fail to disclose all features of the respective claims) of Kazi and Miller would only be possible with the inadmissible use of impermissible hindsight because Kazi relates to the efficient compilation of Java programs and Miller relates to indefinite size intermediate languages. The latter are not

mentioned or referenced in Kazi or not knowledge available to one of ordinary skill in the art, and there would be no reason for the skilled person to combine these two references.

In view of the foregoing, claims 6-10 are further distinguishable over the cited references, individually or in combination.

IV. Rejection Under 35 U.S.C. § 102:

Claims 19-25 are rejected under 35 U.S.C. § 102(e) as being anticipated by Miller et al. (US Patent No. 6,389,590). Claims 19-25 are rejected under 35 U.S.C. § 102(b) as being anticipated by Kazi et al., "Techniques for Obtaining High Performance in Java Program", 7-1999. Claims 19-25 are rejected under 35 U.S.C. § 102(b) as being anticipated by Koizumi et al. (US Patent No. 5,586,323).

Claim 19, as amended, is directed to a distributed computer system comprising a server including a store for storing virtual processor code, said code being a machine-independent representation of the bytecode of an object oriented computer program using an instruction set of a virtual processor, and a plurality of remote client devices in communication with the server, each client device including a client processor, a native translator arranged to translate the virtual processor code into native code which uses the instruction set of the respective client processor, and a native code store; the system including transmission means for transmitting the virtual processor code from the server to the client devices. The bytecode is stack-based, and the virtual processor code is register-based.

Claim 19 has been amended to specify that the byte code is stack-based and the virtual processor code is register-based and that the virtual processor code is a representation of

the byte code of an object or a related computer program. Claim 19 thus corresponds to independent claim 1 as amended.

In these rejections, the Examiner seems to equate virtual processor code with byte code. This is clearly not consistent with the present wording of claim 19, as amended, which explicitly states that it is bytecode which is transformed into virtual processor code and further makes it clear that bytecode is stack-based whereas virtual processor code is register-based. For this and other reasons set forth below, claim 19 and its dependent claims are not anticipated and are distinguishable over each of the cited references.

Turning to Kazi, with respect to independent claim 1, the Examiner acknowledges that Kazi does not disclose step (b) of claim 1, that is transmitting the virtual processor code from a server to a client device. In claim 19, the virtual processor code is stored on the server and is translated into native code by the client. Therefore, the virtual processor code must necessarily be transmitted from the server to the client. However, as acknowledged by the Examiner in relation to claim 1, this feature is not disclosed in Kazi. Thus, claim 19 is not anticipated by Kazi and likewise would not be rendered obvious by the cited references for similar reasons discussed above with reference to claim 1.

Claim 19 also stands rejected as being anticipated by Miller (point 6 of the Office Action). However, there is no disclosure of a virtual processor code which is register-based in Miller. As set out on pages 9 and 10 of the Applicant's last response, the virtual processor code is designed to be closely related to native code for native processors. For example, it is register-based rather than stack-based, as claimed. Miller teaches against this very idea because Miller's intermediate language is designed to handle indefinite sized variables, which requires more rather than less processing for conversion into the native code (it is this extra processing to which

Miller relates). Therefore, Miller actually teaches against the present invention as defined in claim 19.

Finally, claim 19 stands rejected as being anticipated by Koizumi *et al.* However, Koizumi *et al* relates to the translation of source code into an intermediate representation, wherein the claimed invention specifically relates to the translation of bytecode (itself an intermediate representation, see above) into a further intermediate representation, that is virtual processor code. Accordingly, claim 19 is not anticipated by Koizumi.

Furthermore, there is nothing in Koizumi which would motivate one of ordinary skill in the art to translate the source code into a further intermediate representation (bytecode) before translating it into the intermediate representation used in Koizumi. Byte code is already a machine independent intermediate representation which can be executed on various platforms using a Java Virtual Machine. There is no apparent reason why the skilled person would modify Koizumi to introduce an extra intermediate representation. Similarly, if the teaching of Koizumi was applied to Java source code directly, a different intermediate representation would result, one derived directly from the source code rather than the bytecode. Accordingly, claim 19 would also not be obvious in light of Koizumi.

CONCLUSION

Based on the foregoing remarks, the Applicant respectfully requests reconsideration and withdrawal of the rejection of claims and allowance of this application.

AUTHORIZATION

The Commissioner is hereby authorized to charge any additional fees which may be required for consideration of this Amendment to Deposit Account No. 13-4500, Order No. 4362-4002.

In the event that an extension of time is required, or which may be required in addition to that requested in a petition for an extension of time, the Commissioner is requested to grant a petition for that extension of time which is required to make this response timely and is hereby authorized to charge any fee for such an extension of time or credit any overpayment for an extension of time to Deposit Account No. 13-4500, Order No. 4362-4002.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: 6/14/06 By: _____

James Hwa
Registration No. 42,680
(202) 857-7887 Telephone
(202) 857-7929 Facsimile

Correspondence Address:
MORGAN & FINNEGAN, L.L.P.
3 World Financial Center
New York, NY 10281-2101